Performance Analysis of Multi-threaded Multi-core CPUs Format*

Vijayalakshmi Saravanan
VIT University
India
vijayalakshmis@vit.ac.in

Kaushik S
SASTRA University
India
ksanthanam@acm.org

Sai Krishna P
IIT Guwahati
India
k.pallekonda@iitg.ernet.in

D.P Kothari
IIT Delhi
India
dpk0710@yahoo.com

ABSTRACT

Processors are constantly changing and becoming more advanced. They incorporate new concepts and ideas into the architecture with each evolution. One such concept is multi-threading. It aims at increasing the processors performance by reducing its idle time. It is the ability of the processor to execute multiple threads simultaneously on different cores present inside. Multi-threading concepts have also been incorporated in embedded systems which employ either a single-core or multi-core architecture. The aim of this study is to evaluate how effectively multi-threading improves processor utilization on multiple cores by taking both single and dual core processors and evaluating the performance of each by comparing the number of instructions executed per second. The results of this study give an edge to multi-threading in a single-core processor when compared to a dual-core processor when performance aspects are considered. Our analysis helps us to design the processor architecture in such a way that we utilize both the concepts of multi-threading and multi-core architecture to achieve maximum performance. The results of Simultaneous Multi-threading (SMT) performance improvement is encouraging when compared with dual-core processors.

Categories and Subject Descriptors

H.4 [**Computer Architecture**]: Miscellaneous; D.2.8 [**Multi-threaded Multi-core Architecture**]: Metrics—*performance measures*

General Terms

Simulation

Keywords

Performance, Multi-thread, Multi-core and instructions per cycle.

1. INTRODUCTION

The architecture of microprocessors has been constantly evolving. Multi-threading is used to obtain high CPU performance. We see new concepts being introduced every day. We have moved from the early batch processing systems to systems which now support concurrent processing in the form of Multi-threading. Multi-Threading is the process by which the processor is able to execute more than one thread simultaneously and threads are lightweight processes. It aims at increasing the processor utilization by using thread level as well as instruction level parallelism.

In a multi-core architecture, where there is more than one execution unit built inside the processor (called as a core), simultaneous execution takes place. In a single core architecture, this is done in an interleaved manner i.e, switching between threads. Nowadays, embedded devices such as mobile phones and mp3 players are also built along these same lines implementing multi-core architecture. Now the question arises, does a multi-core architecture really produce better performance than a single core processor which implements multi-threading in an interleaved manner? This is the crux of this analysis paper and we have analysed the performance characteristics of both single and dual core processor using an embedded system benchmark suite.

The motivation of this work comes from the fact that multi-core processors are used abundantly everywhere ranging from business, smart phones to server and scientific computing. One of the biggest challenges while moving to multi-core processors is to extract full performance gain provided by multi-core systems. Therefore, we need to thoroughly research on the study of performance bottleneck when using

*(Produces the permission block, and copyright information). For use with SIG-ALTERNATE.CLS. Supported by ACM.

†A full version of this paper is available as *Author's Guide to Preparing ACM SIG Proceedings Using LaTeX2ε and BibTeX* at `www.acm.org/eaddress.htm`

MES '13, June 23 - 24 2013, Tel-Aviv, Israel

SMT and multi-core.
The paper is organized as follows: Section 2 presents some related work. Section 3 describes the overview of proposed approach and the set of benchmarks used for the simulation and work done including the metrics taken for multi-threaded multi-core processors model, and the discussions are presented in Section 4. In Section 5, we conclude our work and talk about future enhancements.

2. RELATED WORK

Prior works discuss on architecture performance modeling. Initially, the processor performance evaluation model studied based on non-uniformity and program parallelism in [1]. The extension of this theoretical performance model is carried out by Noonburg and Shen in [2] and later they proposed statistical framework based on markov chain model [3]. The super-scalar processor models based on queuing theory is discussed in [4].
The problem of power-performance trade offs on multi-core architecture simulation has been intensively investigated in the literature. In [5], Kunkel and Smith studied the pipeline depth and reported its impact on gate delays when considering only the performance metric. V. Srinivasan et. al. consider $BIPS^3/W$ to be the best measure for power-performance as it correlates with the dynamic power consumed which is CV^2f [6]. In [7], Sprangle studied the performance of processor design with respect to the IPC degradation of Intel Pentium 4 processors when adding cycles. Various power-performance metrics were proposed in [8], [6], and these metrics are dependent on the type of processor design. Penolazzi proposed an efficient system level estimation approach at early in the design cycle [9].
Michuad et.al. developed a model based on instruction fetch and branch prediction rate on super-scalar processor performance [10]. In order to quantify the multi-threaded and multi-core processor performance and power, several metrics and methodologies can be explored. In our work, we have focused on architecture-level simulators and processor simulators. In [11], architecture-level simulation techniques for power reduction and performance improvements based on cache and memory organization were discussed. Such representative techniques dealing with energy-efficient based processor design, transistor density, clock frequency, and voltage scaling, have been discussed in [12], [13]. On the other hand, few studies on processor simulators, with focus on performance improvement and power consumption minimization, have been carried out in the literature. In [14], a power model framework based on circuit-level simulators was introduced. In [6], [15], PowerTimer was modeled in such a way that the performance and power can be improved by varying issue-width and pipeline depth. In [16], a comparison of multiprocessors in terms of power and performance was proposed. In [17], the power models of two different architectures were compared and their benefits were reported.
The concept of SMT was introduced into processor design with the release of the Intel Pentium 4 processor. SMT allows multiple independent threads of execution to better utilize the resources provided by modern processor architectures. In simultaneous multi-threading, instructions from more than one thread can be under the process of execution in any given pipeline stage at a time. The number of concurrent threads allowed varied according to the complexity of chip design. Symmetric Multi-Processing (SMP) introduced the use of multiple cores to perform several tasks in parallel, thereby increasing performance. Hence, intuitively SMP based architectures are much more efficient than SMT based processors. But, in practical there is a trade-off between single-core with multi-thread processor model and dual-core multi-thread model.
Even though the above-mentioned proposals on performance modeling have greatly contributed to the understanding of workloads and their performance gains, their analysis have revealed that they do not address the impact of performance while considering multi-threaded model with single-core over SMT with multi-core processor model. This work aims to determine the performance benefits of different processor architectures (such as multi-threaded and multi-core).

3. OVERVIEW OF THE PROPOSED WORK

The proposed approach for multi-threaded multi-core architecture simulation model is depicted in Figure 1. The analysis of multi core architecture using different benchmark suites provides us with a critical view of the performance aspects. But, the question arises- do we really obtain better performance by increasing the number of cores, or can we improve the performance purely by increasing the number of threads available. This comparison between a single core core and dual core architecture opens up this branch of analysis between multi threading and multi core architecture. With this perspective, our objective is to analyze the performance benefits when considering the following scenarios:

- Analyze multi-threaded processor performance by increasing the number of SMT threads in single-core.
- Analyze performance improvement by increasing the number of threads with dual-core.

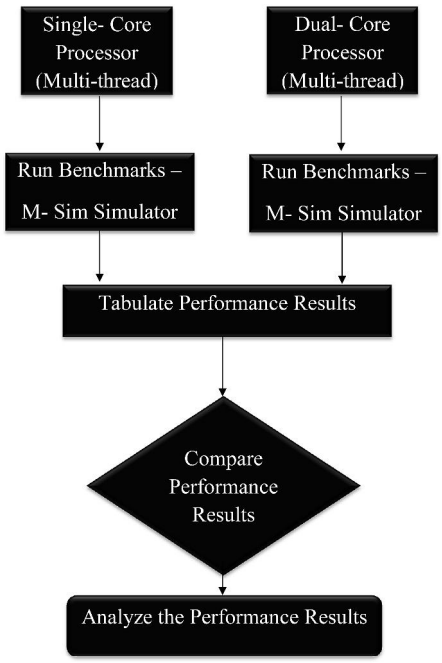

Figure 1: Proposed Approach

The Instructions Per Cycle(IPC) and Billion Instructions Per Second(BIPS) standards provide a performance only measure of multi-core and multi-thread processors as the number of threads are varied on single-core and dual-core respectively. MiBench provides us with a comprehensive graph containing the type of instructions for each argument file. These different categories of applications were taken separately and simulated on both single and dual core processors. The processor parameters are tabulated below.

Table 1: Processor Parameters

Issue Width:8	Window Size : 128
Fetch Width:8	Decode Width:8
Functional Int Units:8 Int ALU	Fuctional FP Units:4 FP ALU
Number of Virtual Registers: 32	Number of Physical Registers: 1024

3.1 Simulator

A literature survey showed that very often the tools used for the analysis are based on embedded applications. Therefore, we decided to use the simulator, M-Sim which is a multi-threaded simulator [18] which uses a modified version of the sim-outorder program of Simple-scalar to support SMT [19]. M-Sim has a detailed cycle-accurate model for key pipeline structures. It supports various features like explicit register renaming, and since it supports execution of multiple threads, it is the most apt simulator available. M-Sim supports unmodified statically linked Alpha AXP binaries. M-Sim has the Wattch tool included which can be later used for considering the power aspects. Though we have various simulation tools available for the power and performance, the advantage of using M-Sim was that we could provide accurate results of SMT and multi-core architecture simulations focused on performance analysis.

3.2 Benchmarks

In order to analyze the selected embedded domain, a set of freely available benchmarks are required. Unfortunately many of the benchmarks are available only commercially and not accessible to researchers in academia. MiBench [20] is built for embedded devices and it provides us with various applications which range from computationally intensive ones to IO intensive. It is targeted at Intel's SA-1 Strong ARM pipeline which are found in the SA-11 xx series of embedded microprocessors. The arguments provided in this suite can be categorized into three types:

- Computationally Intensive
- Memory Intensive
- IO Intensive

This formed an idea of taking such benchmarks for the analysis. The above categories of applications were separately simulated because of the different workload it puts on the CPU. M-Sim accepts statically linked Alpha AXP binaries or EIO traces. Binaries and EIO traces of MiBench workloads such as I/O (bitcount, basicmath, qsort, susan, fft), computational (adpcm, sha, crc32, patricia) and memory intensive benchmarks (rijndael, dijkstra, blowfish) [21]. It should be noted that there are variations in the design level and description due to the different workloads taken for the analysis. We take all these categories of applications and test each type separately on the simulator for both single and dual core processor. The results of the simulation are tabulated.

3.3 Metrics

Given that our work is aimed to analyse the performance, we need to define a set of metrics. Our metric for evaluating performance is instructions executed per second in terms of BIPS (Billion Instructions per Second) defined by Gonzaleze [22]:

- **IPC** : The number of instructions executed per cycle.
- **BIPS**: The number of instructions (in Billion) executed per second (i.e. delay). This is also referred to as peak instruction throughput.

3.4 Simulation

Embedded systems is one field where the processor development is facing exponential growth day by day. We have to consider the aspects of whether the system will truly benefit from having multiple cores in terms of performance considerations. We have taken different categories of application from MiBench, an embedded system benchmark suite and we have run the simulation on both single and dual core processors. A set of benchmarks were run on M-Sim and the number of threads were varied.The threads which are supplied to the simulator are independent threads because M-Sim does not support execution of dependent threads. In Figures, the number of threads are given along x-axis and the BIPS value for each of the various benchmark workloads such computational, IO and memory are given along y-axis. We captured the following parameters through simulation in order to obtain the performance gain: (a) Throughput IPC (b) Time taken (c) Number of instructions executed. Each metric was plotted relatively to the number of varied threads on single and dual-core processors. The results are tabulated in Tables 2 - 4 and results are captured in Figure 3-4.

Table 2: MiBench Computationally Intensive Benchmark Statistics for Single vs. Dual-core

Threads	No. of instructions		Time(in sec)		IPC		BIPS	
0	Single-core	Dual-core	Single-core	Dual-core	Single-core	Dual-core	Single-core	Dual-core
1	1000042	1000042	7	7	0.65	0.65	142863.14	142863.14
2	1420245	2734946	7	17	0.98	1.99	202892.14	160879.17
4	3277001	5912943	21	35	0.99	1.62	156047.66	168941.22
6	2931344	7200556	21	56	0.54	1.29	139587.80	128581.35
8	2825772	7092132	23	63	0.44	0.94	122859.65	112573.52
16	5200147	11404829	57	180	0.47	0.68	91230.64	63360.16

Table 3: MiBench I/O Intensive Benchmark Statistics for Single vs. Dual-core

Threads	No. of instructions		Time(in sec)		IPC		BIPS	
0	Single-core	Dual-core	Single-core	Dual-core	Single-core	Dual-core	Single-core	Dual-core
1	1000042	1000042	5	5	0.81	0.81	200008.4	200008.4
2	1692792	3282973	9	18	0.99	2.17	188088	182387.38
4	3528969	7124675	21	43	1.08	2.18	168046.14	165690.11
6	5314559	10790327	36	75	1.05	2.17	147626.63	143871.02
8	7332957	10549441	54	90	1.08	1.44	135795.5	117216.01
16	7991288	13800845	70	162	0.75	1.90	114161.25	85190.40

Table 4: MiBench Memory Intensive Benchmark Statistics for Single vs. Dual-core

Threads	No. of instructions		Time(in sec)		IPC		BIPS	
0	Single-core	Dual-core	Single-core	Dual-core	Single-core	Dual-core	Single-core	Dual-core
1	1000042	1000042	7	7	0.89	0.89	142863.14	142863.14
2	1819622	3882158	10	23	1.00	1.99	181962.2	168789.47
4	3801140	7717388	24	52	1.00	2.00	158380.83	148411.30
6	5790320	11508448	40	89	1.00	2.00	144758	129308.40
8	7770636	15431980	62	130	1.00	2.00	125332.83	118707.53
16	11785467	13800845	131	162	0.94	1.90	89965.39	85190.40

4. RESULT ANALYSIS AND DISCUSSIONS

A simulation model has been presented to evaluate the instruction throughput performance of the multi-threaded multi-core processor model. From the simulation results a comparison was drawn in Table 5 and the graphs for BIPS were plotted for various types of applications on both single and dual-core processors.

Table 5: Comparison of Various Applications on Single vs. Dual-core

Threads	BIPS(COMP)		BIPS(IO)		BIPS(MEM)		%	%	%
0	Single-core	Dual-core	Single-core	Dual-core	Single-core	Dual-core	COMP	IO	MEM
1	142863.14	142863.14	200008.4	200008.4	142863.14	142863.14	No change	No change	No change
2	202892.14	160879.17	188088	182387.38	181962.2	168789.47	20.7	3.03	7.23
4	156047.66	168941.22	168046.14	165690.11	158380.83	148411.30	-8.26	1.4	6.29
6	139587.80	128581.35	147626.63	143871.02	144758	129308.40	7.88	2.54	10.6
8	122859.65	112573.52	135795.5	117216.01	125332.83	118707.53	8.37	13.68	5.28
16	91230.64	63360.16	114161.25	85190.40	89965.39	85190.40	30.54	25.3	5.3

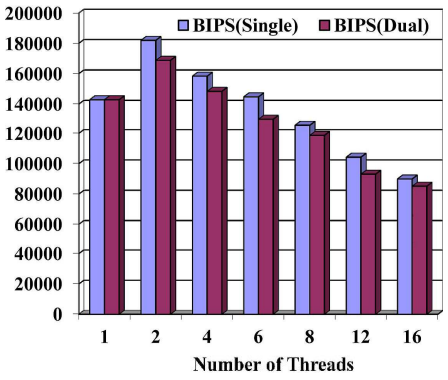

Figure 2: MiBench Memory Intensive Benchmark Results

The main results of our analysis are that:

1. Overall, the simulation results were outperforming well with multi-threaded processors and it shows that the multi-threaded processor model was approximately 13.5 % faster in computational and IO intensive embedded benchmark workloads and 7.5 % faster in memory intensive workloads compared with dual-core processor models which is consistent with its theoretical meaning.

2. From the graphs we can see that considering BIPS which is purely a performance factor that the efficiency is better on multi-threading in single core than multi-threading in dual core processor. It is also noted that the performance may improve if we increase number of cores with one thread in each core (i.e, Multi-core with single-thread).

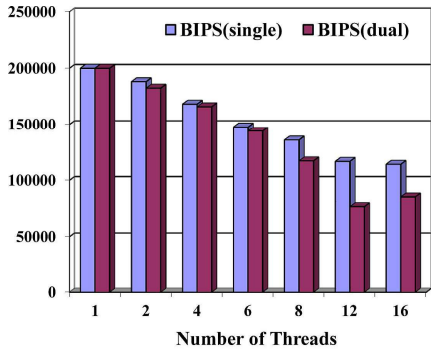

Figure 3: MiBench IO Intensive Benchmark Results

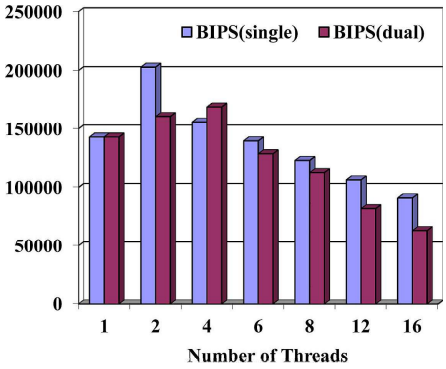

Figure 4: MiBench Computationally Intensive Benchmark Results

5. CONCLUSION

In this study, we examined the effect of performance improvement by increasing the number of threads in SMT and multi-core processor. Based on our study we obtained the following: (1) Among various MiBench workloads, the processor utilization is more efficient in computational applications and memory intensive workloads is least because more the number of threads waiting for memory access, CPU idle time also increases thereby decreasing the performance.(2) Overall, multi-threaded processors with single-core processor model has an advantage over multi-threading in dual-core processors in terms of performance gains and there is no change in performance improvement/degradation on single and dual-core processors with single-thread. In the future we can expand this study to include power aspects so that we can conclusively arrive at a power-performance tradeoff between single and dual core processors. Also, we can modify existing multi-core schedulers which keep all the cores busy to the maximum extent possible.

6. REFERENCES

[1] N. P. Jouppi, The nonuniform distribution of instruction-level and machine parallelism and its effect on performance, IEEE Trans. Comput. 38 (12) (1989) 1645–1658. doi:10.1109/12.40844.
URL http://dx.doi.org/10.1109/12.40844

[2] D. B. Noonburg, J. P. Shen, Theoretical modeling of superscalar processor performance, in: Proceedings of the 27th annual international symposium on Microarchitecture, MICRO 27, 1994.

[3] D. B. Noonburg, J. P. Shen, A framework for statistical modeling of superscalar processor performance, in: Proceedings of the 3rd IEEE Symposium on High-Performance Computer Architecture, HPCA '97, 1997.

[4] Y. Zhu, W. F. Wong, Sensitivity analysis of a superscalar processor model, Aust. Comput. Sci. Commun. 24 (3) (2002) 109–118.

[5] S. R. Kunkel, J. E. Smith, Optimal pipelining in supercomputers, SIGARCH Comput. Archit. News 14 (1986) 404–411.

[6] D. Brooks, P. Bose, V. Srinivasan, M. K. Gschwind, P. G. Emma, M. G. Rosenfield, New methodology for early-stage, microarchitecture-level power-performance analysis of microprocessors, IBM J. Res. Dev. 47 (2003) 653–670.
doi:http://dx.doi.org/10.1147/rd.475.0653.

[7] E. Sprangle, D. Carmean, Increasing processor performance by implementing deeper pipelines, in: Proceedings of the 29th International Symposium on Computer Architecture (ISCA-29), 2002.

[8] V. Zyuban, P. Strenski, Unified methodology for resolving power-performance tradeoffs at the microarchitectural and circuit levels, in: Proceedings of the 2002 international symposium on Low power electronics and design, ISLPED '02, 2002, pp. 166–171.

[9] S. Penolazzi, A system-level framework for energy and performance estimation in system-on-chip architectures, Ph.D. thesis, KTH, Electronic Systems, qC 20110315 (2011).

[10] P. Michaud, A. Seznec, S. Jourdan, An exploration of instruction fetch requirement in out-of-order superscalar processors, Int. J. Parallel Program. 29 (1) (2001) 35–58.

[11] Y. Meng, T. Sherwood, R. Kastner, Exploring the limits of leakage power reduction in caches, ACM Trans. Archit. Code Optim. 2 (3) (2005) 221–246.

[12] J. Pouwelse, K. Langendoen, H. Sips, Dynamic voltage scaling on a low-power microprocessor, in: Proceedings of the 7th annual international conference on Mobile computing and networking, MobiCom '01, 2001, pp. 251–259.

[13] V. Venkatachalam, M. Franz, Power reduction techniques for microprocessor systems, ACM Comput. Surv. 37 (3) (2005) 195–237.

[14] M. Moudgill, J.-D. Wellman, J. H. Moreno, Environment for PowerPC Microarchitecture Exploration, IEEE Micro 19 (1999) 15–25. doi:10.1109/40.768496.

[15] Y. Li, B. Lee, D. Brooks, Z. Hu, K. Skadron, CMP design space exploration subject to physical constraints, in: Proceedings of the 12th International Symposium on High Performance Computer Architecture, 2006.

[16] A. Hyari, A comparative study on heterogeneous and homogeneous multiprocessors, Ph.D. thesis (December 2009).
URL http://www.abandah.com/gheith/Courses/CPE731_F09/Research_Projects/5_Report.pdf

[17] S. Ghiasi, A Comparison of Two Architectural Power Models, in: Workshop on Power-Aware Computer Systems, 2000, pp. 137–152.

[18] K. G. Joseph J. Sharkey, Dmitry Ponomarev, Abstract M-SIM: A Flexible, Multithreaded Architectural Simulation Environment, Tech. rep. (2005).

[19] T. Austin, A Userś and Hackerś Guide to the SimpleScalar Architectural Research Tool Set, Website (1997).
URL http://www.cs.virginia.edu/~skadron/cs654/slides/hack_guide.pdf

[20] S. M. Z. Iqbal, Y. Liang, H. Grahn, Parmibench - an open-source benchmark for embedded multiprocessor systems, IEEE Comput. Archit. Lett. 9 (2) (2010) 45–48. doi:10.1109/L-CA.2010.14.

[21] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, R. B. Brown, Mibench: A free, commercially representative embedded benchmark suite, in: Proceedings of the Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop, WWC '01, 2001, pp. 3–14. doi:10.1109/WWC.2001.15.

[22] R. Gonzalez, M. Horowitz, Energy Dissipation In General Purpose Microprocessors, IEEE Journal of Solid-State Circuits (1996) 1277–1284.